INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An efficient method for the coordinate transformation problem of massively three-dimensional networks

Károly Németh — Olivier Coulaud — Gérald Monard — János G. Ángyán

N° 4083

Décembre 2000

THÈME 4

ISSN 0249-6399 ISRN INRIA/RR--4083--FR+ENG

An efficient method for the coordinate transformation problem of massively three-dimensional networks

Károly Németh*†, Olivier Coulaud , Gérald Monard* , János G. Ángyán*

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Numath

Rapport de recherche n° 4083 — Décembre 2000 — 16 pages

Abstract: A new and efficient algorithm is presented for the coordinate transformation problem of massively three-dimensional networks formed e.g. by the atoms of crystal fragments or molecular clusters. The new algorithm is based on a divide-and-conquer technique to perform iterative coordinate transformation, applicable even for three-dimensional networks, with linear scaling memory and near linear scaling CPU time requirements. The new algorithm proved to be very fast in the coordinate transformation problems and geometry optimization of diamond fragments, water clusters, globular proteins and solvated proteins.

Key-words: coordinate transformation, internal coordinate, linear scaling, geometry optimization

* Groupe de Chimie théorique, UMR CNRS 7565, Université Henri Poincaré, B.P. 239, F-54506 Vandœuvre-lès-Nancy, France
† Centre Charles Hermite, B.P. 239, F-54506 Vandœuvre-lès-Nancy, France

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 —Télécopie : +33 3 83 27 83 19

Une méthode efficace pour la transformation de coordonnées dans de très gros système moléculaire

Résumé : Nous décrivons un nouvel algorithme efficace pour effectuer le changement de coordonnées entre les coordonnées cartésiennes et les coordonnées internes dans de très gros système moléculaire formés par des fragments de cristaux ou des clusters de molécules. Cet algorithme, basé sur une technique de "diviser pour régner", réalise ce changement de coordonnées en temps et en occupation mémoire linéaire en fonction du nombre d'atomes. Nous montrons que l'algorithme est très rapide pour l'optimisation de géométrie de fragments de diamants, de clusters d'eau, de protéines globulaires et pour des protéines dans un solvant.

Mots-clés : coordonnées internes, méthode à croissance linéaire, optimisation géométrique, transformation de coordonnées

1 Introduction

Many important developments have been made recently in the numerical treatment of the coordinate transformation problem of molecular geometry optimization. These developments have been motivated essentially by the rapid evolution of linear scaling electronic structure calculation techniques recently reviewed by Goedecker [1] and Scuseria [2]. Farkas and Schlegel constructed an algorithm whose CPU time consumption scales quadratically ($\mathcal{O}(N^2)$) with the system size (N) [3]. Billeter, Turner and Thiel developed a linear scaling ($\mathcal{O}(N)$) algorithm which uses hybrid delocalized coordinates in the framework of a divide-and-conquer method [4]. Paizs, Baker, Fogarasi, Suhai and Pulay use the preconditioned conjugate gradient method in conjunction with an adaptive Cholesky decomposition [5, 6]. Kudin, Scuseria and Schlegel extended the applicability of internal coordinates to the optimization of periodic systems [7]. In a recent article, we used the Newton-Raphson method, spectral shift and incomplete Cholesky factorization with symmetric permutations to carry out the necessary transformations between the Cartesian and internal coordinate representation of vectors [8].

In the present article we consider the coordinate transformation problem in the special case of massively three-dimensional (3D) networks. Such networks are formed, e.g. by the primitive internal coordinates of crystal fragments, clusters of molecules or proteins surrounded by solvent molecules. Since, up to our knowledge, this case has not yet been carefully studied in the literature, we focus our attention to this subject now.

Cholesky factorization plays an important role, not only in coordinate transformations but also in other fields of computational chemistry, for example in the transformation of Fock matrices from non-orthonormal into orthonormal representation. For this transformation, Cholesky factorization was first used by Millam and Scuseria [9]. It achieves linear scaling for quasi one-dimensional molecules [9], but scales only quadratically for 3D networks. Challacombe proposed a linear scaling technique for this latter case [10], using the AINV algorithm, developed by Benzi, Meyer and Tuma [11].

Here, we suggest to combine Cholesky factorization with a special divide-and-conquer technique to address the challenge of coordinate transformation of massively 3D networks. The paper is built up as follows. First we briefly summarize our previously described algorithm [8] particularly adapted to treat coordinate transformations of quasi one-dimensional (1D) systems. Then we show, how to divide 3D networks into 1D subsystems and solve the 3D problem by combining results from the solutions of the 1D subsystems. We illustrate the performance of our method on large diamond fragments, water clusters and globular proteins. Finally we summarize the new results presented in the paper.

2 Algorithm for quasi 1D systems

In geometry optimizations performed in internal coordinates, Cartesian energy gradients, $\mathbf{g_c}$ must be transformed into internal coordinate representation, $\mathbf{g_i}$. Then, the displacements of the $\mathbf{q}$ internal coordinates, $\mathbf{\Delta q}$ is constructed from the internal gradients. These displace-

ments must be transformed back into Cartesian ones ($\mathbf{\Delta x}$) and used to update the Cartesian coordinates ($\mathbf{x}$) of the molecule.

The gradient transformation is formulated by

$$\mathbf{g_i} = \mathbf{g_c}\mathbf{A}. \tag{1}$$

Here, and in the following developments all vectors denote row vectors. The matrix $\mathbf{A}$ is the Morse-Penrose pseudoinverse of the vibrational $\mathbf{B}$ matrix [12, 13]. The transformation of coordinate displacements from internal to Cartesian representation happens via the iterative back-transformation

$$\mathbf{x^{(k+1)}} = \mathbf{x^{(k)}} + (\mathbf{q} - \mathbf{q^{(k)}})\mathbf{A^t}. \tag{2}$$

$\mathbf{q}$ contains the required new values of the internal coordinates, while $\mathbf{q^{(k)}}$ holds their value in the kth iteration step. The upper index t denotes matrix transposition. The iterative back-transformation is a Newton algorithm, which is applied because of the nonlinear nature of the relationship between the Cartesian and internal coordinates. The $\mathbf{B}$ matrix is constructed from the derivatives of the internal coordinates according to Cartesian ones:

$$B_{kl} = \frac{\partial q_k}{\partial x_l}, \tag{3}$$

where q_k is the kth internal coordinate and x_l is the lth Cartesian coordinate. $\mathbf{B}$ is always sparse provided that local, e.g. primitive, internal coordinates are used. $\mathbf{A}$ is constructed traditionally as the right-hand-side pseudoinverse of $\mathbf{B}$, i.e. $\mathbf{A}(\mathbf{B}\mathbf{B^t}) = \mathbf{B^t}$. However, as it has been pointed out by several authors, the left- and right-hand-side pseudoinverses of $\mathbf{B}$ are equivalent, and it is less elaborate to work with the left-hand-side one [3, 6, 8]. We consider $\mathbf{A}$, the left-hand-side pseudoinverse, given by

$$\mathbf{G_c}\mathbf{A} = \mathbf{B^t}, \tag{4}$$

with

$$\mathbf{G_c} = \mathbf{B^t}\mathbf{B}. \tag{5}$$

$\mathbf{G_c}$ is usually singular, with 6 zero eigenvalues, therefore its inverse must be constructed in a generalized sense, by inverting only the nonzero eigenvalues of the spectral resolution. $\mathbf{B}$ and $\mathbf{A^t}$ are general rectangular, $N_i \times N_c$ matrices, while $\mathbf{G_c}$ is of the size $N_c \times N_c$. N_c denotes the number of Cartesian coordinates: $N_c = 3N$, where N is the number of atoms. N_i denotes the number of internal coordinates, which is usually much larger than N_c.

$\mathbf{G_c}$ is positive semidefinite by construction. The singularity of $\mathbf{G_c}$ can be eliminated by a small spectral shift:

$$\mathbf{G}_\varepsilon = \mathbf{G_c} + \varepsilon\mathbf{I_c}, \tag{6}$$

where $\mathbf{I_c}$ is the $N_c \times N_c$ identity. As we have pointed out, the generalized inverse of $\mathbf{G_c}$ can be approximated by the inverse of $\mathbf{G}_\varepsilon$ [8]. According to our experience, this approximation

works very well in the coordinate transformation problem. Multiplication of vectors by the inverse of $\mathbf{G}_\varepsilon$ can be carried out efficiently via the $\mathbf{U}_\varepsilon$ Cholesky factors of $\mathbf{G}_\varepsilon$, which are defined by

$$\mathbf{G}_\varepsilon = \mathbf{U}_\varepsilon^{\mathbf{t}} \mathbf{U}_\varepsilon. \tag{7}$$

Using the quasi Newton algorithm, the gradient transformation problem can be solved in an iterative fashion

$$\mathbf{g_i^{(k+1)}} = \mathbf{g_i^{(k)}} + [\mathbf{g_c} - \mathbf{g_i^{(k)}} \mathbf{B}] \mathbf{U}_\varepsilon^{-\mathbf{1}} \mathbf{U}_\varepsilon^{-\mathbf{t}} \mathbf{B^t}, \tag{8}$$

with $\mathbf{g_i^{(0)}} = \mathbf{0}$. This iteration converges to a solution of

$$\mathbf{g_c} = \mathbf{g_i} \mathbf{B}, \tag{9}$$

which is equivalent to Eqn (1). It is important to note that there is, in principle, an infinite number of solutions to this equation, since any vector of the zero subspace of the $\mathbf{G_i} = \mathbf{B}\mathbf{B^t}$ matrix can be added to a solution of Eqn (9), $\mathbf{g_i}$, and the sum still remains a solution of the same equation.

The explicit evaluation of the inverse Cholesky factors is not necessary, since we are interested only in the results of multiplications by these inverses. Such multiplications can be carried out very efficiently by using forward and backward substitutions, thus avoiding the explicit evaluation of the non-sparse inverses of the Cholesky factors.

The iterative back-transformation can be formulated similarly:

$$\mathbf{x^{(k+1)}} = \mathbf{x^{(k)}} + (\mathbf{q} - \mathbf{q^{(k)}}) \mathbf{B} \mathbf{U}_\varepsilon^{-\mathbf{1}} \mathbf{U}_\varepsilon^{-\mathbf{t}}. \tag{10}$$

As already mentioned, $\mathbf{B}$ is very sparse when local internal coordinates are used. This ensures also high sparsity in $\mathbf{G}_\varepsilon$. However, even if $\mathbf{G}_\varepsilon$ is very sparse there is no guarantee that the corresponding $\mathbf{U}_\varepsilon$ Cholesky factors become also sparse. For quasi one-dimensional systems $\mathbf{G_c}$ can be brought to an approximately banded structure, or to a tree-type structure by symmetric permutations. For chain type molecules, with no spatial connection between topologically far parts of the chain, this permutation can be best done by minimum degree ordering. If there are, in addition, some side chains the optimal ordering can be carried out by nested dissection technique. By these permutations the profile of the $\mathbf{G_c}$ matrix gets minimized [14]. All nonzero elements of the corresponding Cholesky factors appear inside the profile. However, there are systems in which even reordering of $\mathbf{G_c}$ by symmetric permutations cannot help to make the profile as simple as in the above mentioned cases. Such systems are formed e.g. by massively three-dimensional connectivity networks which appear in crystals, in clusters of molecules or in globular proteins surrounded by solvent. At this point, it is important to emphasize that the structure of $\mathbf{G_c}$ reflects the connectivity of the atoms by the internal coordinates, and the sparsity of $\mathbf{G_c}$ is not a sufficient condition for a linearly scaling Cholesky factorization. Considering a series of $\mathbf{G_c}$ matrices constructed for massively 3D networks of increasing size, but having similar topological structure, the

number of nonzero elements of the corresponding Cholesky factors will increase quadratically with the number of atoms. As a consequence, memory and CPU requirements for the Cholesky factorization scale quadratically with system size. This phenomenon is very well known in the theory of Cholesky factorization [14], and has also been recognized in works of Scuseria [9] and Challacombe [10].

It follows from the above discussion that the previously described algorithm [8] should be further improved to handle the coordinate transformation problem in a linear scaling fashion for 3D networks.

3 Algorithm for 3D networks

3.1 Conceptual formulation

In order to achieve linear scaling, at least in the memory requirements, we need to reduce the complexity of the $\mathbf{G_c}$ matrices to be factorized. This is equivalent to a reduction of the connectivities inside the molecule, in other words, we have to remove some connectivities (internal coordinates) and do the transformations only on the rest. Since all internal coordinates are important for the optimization of the molecule, in a forthcoming step of the coordinate transformation procedure we cut another set of internal coordinates and include previously cut ones into the transformation. Thus, during the coordinate transformation procedure we let to relax components of the internal gradients to a solution of Eqn (1) in a sequence, and we do similarly for the iterative back-transformation. This kind of methodology to solve large systems of equations is called in general divide-and-conquer technique. Following these lines we may write for the gradient transformation:

$$\mathbf{g_i^{(k+1)}} = \mathbf{g_i^{(k)}} + [\mathbf{g_c} - \mathbf{g_i^{(k)}}\mathbf{B}]\mathbf{U_{r,k}^{-1}}\mathbf{U_{r,k}^{-t}}\mathbf{B_{r,k}^{t}}, \tag{11}$$

and similarly for the coordinate transformation:

$$\mathbf{x^{(k+1)}} = \mathbf{x^{(k)}} + [\mathbf{q} - \mathbf{q^{(k)}}]\mathbf{B_{r,k}}\mathbf{U_{r,k}^{-1}}\mathbf{U_{r,k}^{-t}}. \tag{12}$$

In these equations $\mathbf{B}$ contains contributions from all internal coordinates, $\mathbf{B_{r,k}}$ has no contributions from the removed internals, $\mathbf{U_{r,k}}$ is constructed by the factorization of $\mathbf{G_{r,k}} + \varepsilon\mathbf{I_c}$ with $\mathbf{G_{r,k}} = \mathbf{B_{r,k}^{t}}\mathbf{B_{r,k}}$. $\mathbf{B_{r,k}}$ and $\mathbf{U_{r,k}}$ change from iteration to iteration.

3.2 Prism distributions

In order to determine $\mathbf{B_{r,k}}$, we have to decide which internal coordinates be active and which be inactive at different steps of the coordinate transformation. Such a choice could be based on a topological analysis of the molecule. We want to avoid performing a general topological analysis, but still to have a generally applicable algorithm. A simple alternative is to divide the molecule into a system of non-overlapping prisms, and then consider only those internal coordinates, which are not cut by the surfaces of these prisms. The choice of prisms has the

advantage that the molecule is allowed to make the maximum possible relaxation along the whole length of the prisms, i.e. along the full size of the molecule. This feature would not be available in case of choosing cubes. For different iteration steps of the coordinate transformation different distributions of prisms may be used. In the construction of these prism distributions there are two main points which must be considered. First, all internal coordinates must appear at least once unremoved. The second point is that maximum available relaxation should be made possible in all coordinate directions. A minimum of five prism distributions can satisfy these conditions. Currently we use six different prism distributions, in order to allow sufficient relaxation in all coordinate directions. These prism distributions consist of uniform prisms. For each coordinate direction two differently positioned distributions of prisms are used. The relative positions of all these prism distributions are set such that they together satisfy the first condition. Furthermore, limited interaction between the prisms may also be allowed. This interaction is switched on by allowing some internal coordinates to be active, if these coordinates connect the tops or bottoms of spatially and topologically neighboring prisms. Thus the isolated prisms become linked and form a quasi one-dimensional system. The different prism distributions are used sequentially in different iteration steps of the coordinate transformation. The cross-section of each prism is kept small, independent from the length of the prism, which latter is of the system size. The size of the cross section is determined by the length of the longest connectivity established by local internal coordinates. Usually, at least twice this length is applied to set the size of the smaller edges of the prisms. If some of the internal coordinates do not fit into the prism distributions, the default size of the prisms can automatically be readjusted, before any matrix operations happen.

Inside the prisms, the atoms are numbered such that for the linked system of prisms, which forms a quasi one-dimensional structure, the numbering starts at one end of the 1D system and ends at the other end, scanning continuously the line which defines one-dimensionality. The $\mathbf{G}_{\mathbf{r},\mathbf{k}}$ matrices of these systems, as well as the corresponding Cholesky factors will always be band diagonal.

We have found that there is a big difference in the number of iteration steps when using linked or unlinked prisms. For linked prisms the gradient transformation converges often in 3-4 steps, thus some of the prism distributions will not be used in the evaluation of internal gradients. This results in a special solution of Eqn (9), namely no forces will be generated on some internal coordinates. The physical meaning of such a solution is that not all internal coordinates are needed for the optimization of the molecule. However, for the stability of the iterative back-transformation and the geometry optimization it is usually much better to use all internal coordinates. In order to have forces on all internal coordinates, in the first six iteration steps non-linked prisms are used, thus avoiding too rapid convergence. Then, in the forthcoming iteration steps linked prisms are applied to accelerate convergence. For the iterative back-transformation only linked prisms are applied.

3.3 Choice of internal coordinates

The above procedure works very well for systems like diamond fragments or water clusters. For water clusters, beside the local primitive internal coordinates, further distance coordinates have to be introduced in order to describe water-water interaction. For globular proteins, in order to achieve rapid convergence in the coordinate transformations, extra internal coordinates need to be introduced beside the primitive ones. In these systems, isolated fragments of the molecules may turn up inside some of the prisms. Thus, the relative position of these fragments is very poorly described. As a result, the coordinate transformation may not converge. A solution is to introduce extra internal coordinates, which bridge the neighboring fragments inside the prisms. Thus we may increase the number of nonzero elements inside the band of $\mathbf{G_{r,k}}$, without generating any new elements outside the band. Correspondingly, the Cholesky factors remain band-diagonal and the memory and CPU time requirements of the factorization scale linearly with system size.

We generate the extra internal coordinates by the following means. First, the prisms are divided into approximately uniform boxes each of them containing at least 3 atoms. The length of these boxes is equal to the width of the prisms. If a box contains less then 3 atoms, it is merged to one of its neighbors. Then, the largest surface triangle formed by the atoms in the box is determined in all the boxes. The atoms of the triangle are connected by extra stretchings. Then all other atoms of the box are connected to the triangle atoms, each by three extra stretchings. The triangle atoms of neighboring boxes are also linked by three stretchings. After the introduction of these new coordinates rapid convergence can be achieved also in the coordinate transformations of globular proteins. We call these new coordinates skeletal coordinates. Obviously, the problem of appearance of isolated fragments in the prisms could have been treated by various choices of internal coordinates, but an exhaustive analysis of the different possibilities is out of the scope of the present paper.

As an alternative solution to the above problems we have also tried to include the complete set of Cartesian displacement coordinates beside the primitive internal ones. However, it did not help much to decrease the number of iteration steps so far. The reason is probably that the relative position of the isolated fragments can be optimized only slowly in Cartesian coordinates. It is still worth having a look at the formalism of these coordinate transformations since it gives a physical interpretation of the spectral shift we apply on the $\mathbf{G_c}$ matrices. The $\mathbf{B}$ matrix of the Cartesian displacement coordinates is just the $\mathbf{I_c}$ identity. Thus, after the inclusion of the Cartesian displacements, the modified $\mathbf{B}$ matrix becomes $\mathbf{B^* = B \oplus I_c}$ and the corresponding $\mathbf{G_c}$ matrix $\mathbf{G_c^* = G_c + I_c}$ is always positive definite. The gradient transformation thus becomes

$$\mathbf{g_i^{(k+1)} \oplus g_c^{*(k+1)} = g_i^{(k)} \oplus g_c^{*(k)} + [g_c - (g_i^{(k)} \oplus g_c^{*(k)})B^*]G_c^{*-1}B^{*t}}. \tag{13}$$

$\mathbf{g_c^{*(k)}}$ is allowed to converge to a different vector than what $\mathbf{g_c}$ is. This flexibility must be allowed, since $\mathbf{B^*}$ may mix the Cartesian and internal components of $\mathbf{g_i^{(k)} \oplus g_c^{*(k)}}$. In the light of the above expressions it is obvious that the spectral shift technique [Eq. (6)]

is equivalent to the inclusion of the Cartesian coordinates with a small weight. Since the weighting factor, ε, is chosen to be quite small, the resulting Cartesian component $\mathbf{g_c^{*(k)}}$ will practically be zero. Therefore it is not necessary to evaluate it explicitly.

3.4 Projecting out redundancy, translations and rotations

According to our experience, if a highly redundant set of internal coordinates is used and the displacements of internal coordinates are small, the iterative back-transformation may not converge. This behavior is due to the redundancy of internal coordinates. The internal coordinate displacements are supposed to lie in a N_f dimensional space, where N_f is $3N-6$ for general molecular structures and $3N-5$ for linear molecules. However, if a redundant set of internal coordinates is used, this property of the displacements is usually not ensured. The space, in which the nonredundant components of the displacements are lying is defined by the columns of the $\mathbf{B}$ or by the rows of the $\mathbf{A}$ matrix. Working with $\mathbf{B_{r,k}}$ matrices implies that the definition of this space varies from iteration to iteration. The required displacement of the internal coordinates, $\mathbf{\Delta q}$, can be decomposed into two parts. One of them, the non-redundant one, $\mathbf{\Delta q_{nr}}$, is lying in the space spanned by the columns of $\mathbf{B}$, the redundant one, $\mathbf{\Delta q_r}$, is lying out of this space. The iterative back-transformation acts only on the non-redundant component, thus the redundant one cannot be eliminated by the iterations. The convergence of the back-transformation is detected by the fact that the Cartesian coordinates do not change, even if the required change of the internals is not completely satisfied. This means that only the non-redundant component of $\mathbf{\Delta q}$ has been eliminated. However, if different $\mathbf{B_{r,k}}$ matrices are used, the space defining the non-redundant component also changes and oscillations may happen during the back-transformation. We have found that these convergence problems can be avoided, if only the non-redundant component of $\mathbf{\Delta q}$ is used, instead of the total $\mathbf{\Delta q}$ in the iterations, and $\mathbf{\Delta q_{nr}}$ is determined by the complete $\mathbf{B}$ matrix. Paizs *et. al.* constructed [6] a projector, which separates the redundant and non-redundant components. For building this projector, $3N-6$ rows of the $\mathbf{B}$ matrix must be selected and their overlap matrix must be built. Then this $(3N-6)\times(3N-6)$ overlap matrix is Cholesky factorized and used in the projection. This technique is very similar to the rank revealing Cholesky factorization [15]. However, as Paizs *et. al.* mention [6], carrying out the projection may need some experimentation. The instability of such procedures is well known from the theory of rank revealing Cholesky factorization [15]. Also, in many cases the factorization of an overlap-type matrix scales quadratically with the system size, as discussed above. To overcome these limitations, we constructed a different algorithm to project out the non-redundant component of $\mathbf{\Delta q}$. We use the complete $\mathbf{B}$ matrix and the matrices we invert are all very small. Our projection scheme is very fast, stable and can efficiently be applied also for massively 3D networks.

We select out columns of the $\mathbf{B}$ matrix sequentially, maximum 100 columns are selected at a time, and they are stored in the matrix $\mathbf{B_k}$. Then we build the corresponding projector, $\mathbf{P_k}$ to the space spanned by the selected columns. This projector acts on $\mathbf{\Delta q}$. The vector, $\mathbf{\Delta q_{nr}^{(k)}}$, which has been projected out, is stored in a buffer. $\mathbf{\Delta q}$ will be modified

by subtracting $\mathbf{\Delta q}_{\mathbf{nr}}^{(\mathbf{k})}$. Then the next projector is constructed from the next columns of $\mathbf{B}$ and acts on the actual $\mathbf{\Delta q}$. The result is added to the buffer and eliminated from $\mathbf{\Delta q}$. Thus, even if non-orthogonal projectors are used, the non-redundant and redundant components can be separated, since the modified $\mathbf{\Delta q}$-s will not have components from the spaces used in previous projections. The projection technique can be summarized by the following expressions:

$$\mathbf{\Delta q}_{\mathbf{nr}}^{(\mathbf{k+1})} = \mathbf{\Delta q}_{\mathbf{nr}}^{(\mathbf{k})} + \mathbf{P}^{(\mathbf{k})}\mathbf{\Delta q}^{(\mathbf{k})}, \tag{14}$$

$$\mathbf{\Delta q}^{(\mathbf{k+1})} = \mathbf{\Delta q}^{(\mathbf{k})} - [\mathbf{\Delta q}_{\mathbf{nr}}^{(\mathbf{k+1})} - \mathbf{\Delta q}_{\mathbf{nr}}^{(\mathbf{k})}], \tag{15}$$

$$\mathbf{P}^{(\mathbf{k})} = \mathbf{B_k}\mathbf{G_k^{-1}}\mathbf{B_k^t}, \tag{16}$$

with

$$\mathbf{G_k} = \mathbf{B_k^t}\mathbf{B_k}. \tag{17}$$

The matrix $\mathbf{G_k}$ may be singular, thus its inverse must be calculated by singular value decomposition (SVD). However, since its size is kept at a fixed small value, performing the SVD is easy. $\mathbf{\Delta q}^{(\mathbf{0})}$ equals with $\mathbf{\Delta q}$ given by the minimizer. Multiplications by the $\mathbf{P}^{(\mathbf{k})}$ projector can most efficiently be carried out by sequential multiplications by the projector components.

Similarly, rotations and translations of the whole molecule must be projected out from the displacements of the Cartesian coordinates. This can be done via constructing the vectors of molecular translations and infinitesimal rotations. In our experience, translational and rotational components of the Cartesian displacements turned out to be very small.

4 Results and discussion

4.1 Computational details

For all sparse matrix algebraic operations a zero threshold of 10^{-10} atomic units (a.u.) has been applied. The spectral shift has been chosen to $\varepsilon = 10^{-6}$ a.u.. The following test systems have been used. Diamond fragments of cubic shape, which have been generated by translating the elementary cell of diamond. Cubic clusters of water molecules. Biomolecular coordinates have been taken from The Protein Data Bank (PDB) [16]. Crambin (PDB entry 1CBN, 642 atoms), Bovine Pancreatic Trypsin Inhibitor (1BPI, 898 atoms), Urate Oxidase (1UOX, 4684 atoms) and Beta-Lactamase (4BLM, 4023 atoms). Some of this biomolecules have also been studied in spherical water environment. The complete set of primitive internal coordinates has been applied in all calculations. For water clusters, in addition to the primitive coordinates, distance coordinates have been introduced. To all biomolecules skeletal coordinates have been added, as defined above. The Cholesky

factors have been recalculated at each iteration step of the transformation. The prisms have been ≈ 14 a.u. wide (twice the length of the elementary cell of diamond). During the iterative back-transformation, first convergence is achieved for a given prism-distribution, then the back-transformation proceeds similarly on the next prism distribution. After all six distributions of prisms are finished, back-transformation restarts with the first one. Global convergence is achieved, when the maximum change of the geometry is smaller than 0.001 a.u. for all prisms.

The algorithm has been implemented in Fortran77 and attached to the GAUSSIAN suit of programs [17]. Energies and Cartesian gradients have been evaluated by means of the DREIDING force field [18]. All calculations have been carried out on a single MIPS R10000 (180 MHz) processor of a SGI PowerChallenge computer. The gradient transformations have been considered as converged, when the maximum correction of $\mathbf{g_i^{(k)}}$ has become smaller than $5 \cdot 10^{-5}$ a.u. for all prism distributions. In all steps of the iterative back-transformation the $\mathbf{B}$ matrix is recalculated. For geometry optimization the steepest descent method has been used, in order to save memory. During the coordinate transformations all matrices have been recalculated at each iteration steps.

4.2 Matrix sparsity

Fig (1) shows the number of nonzero elements in the Cholesky factors (NZ) of different diamond and water cluster systems. The largest molecules, which have been used in the tests contain about 12000 atoms. Since there are six different prism distributions for all molecules, Fig (1) shows all corresponding NZ-s. It can be clearly seen that the number

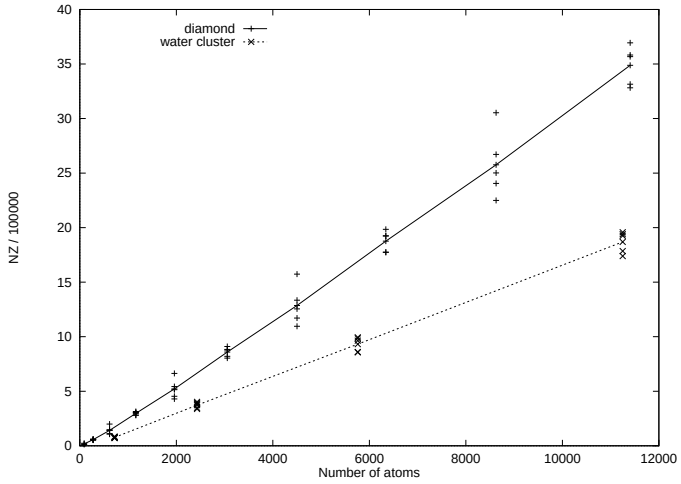

Figure 1: Number of nonzero elements in the lower triangle of the Cholesky factors calculated for diamond fragments and water clusters at different prism distributions.

of nonzero elements of the Cholesky factors scales linearly with system size, while some prism distributions result in sparser matrices than the others. Other matrices, used in the transformations are much sparser than the Cholesky factors. Linear scaling of the number of nonzero elements in the Cholesky factors is a result of the quasi one-dimensionality introduced by the divide-and-conquer scheme via the prism distributions.

4.3 Number of iteration steps

The number of iteration steps in the gradient transformation does not change much with the system size for diamond. For the smallest diamond fragments (about 100 atoms) it is 7, for the largest ones (about 12000 atoms) it is 10. For water clusters this number increases from 14 (smallest cluster, about 600 atoms) to 28 (largest cluster, 12000 atoms). This number is strongly influenced by the choice of internal coordinates. It is also worth mentioning that if there are atoms, which remain completely isolated in any of the prisms, the number of iteration steps may strongly increase. A tool to avoid this problem is the introduction of skeletal coordinates. The number of iteration steps in the iterative back-transformation was not larger than 12 for the systems studied in the present work.

4.4 CPU times

Fig (2) shows average CPU times required for one step of the gradient transformation of diamond and water cluster systems. The CPU times scale linearly with system size. For

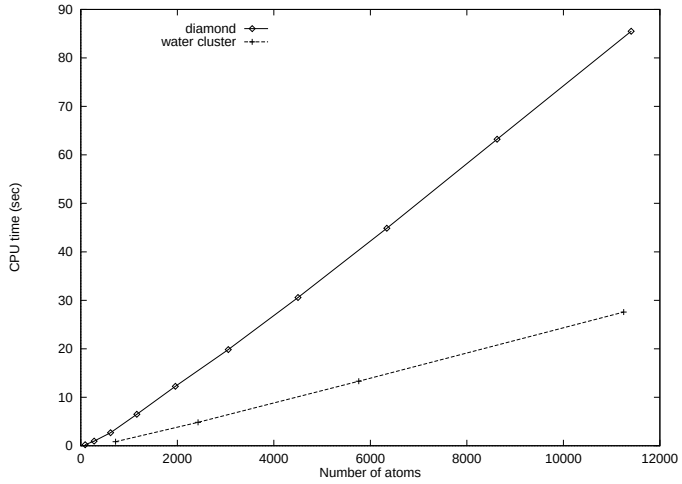

Figure 2: CPU times (sec) of one gradient transformation step for diamond fragments and water clusters.

comparison, total CPU times of the gradient transformations of diamond fragments are

shown in Fig (3). Data were collected from calculations with prism distributions and from calculations without prisms, using nested dissection ordering. It can be clearly seen that nested dissection requires quadratically scaling CPU times, while the divide-and-conquer technique scales near linearly. The same holds also for the memory requirements.

One step of the iterative back-transformation requires a similar amount of CPU time than one step of the gradient transformation.

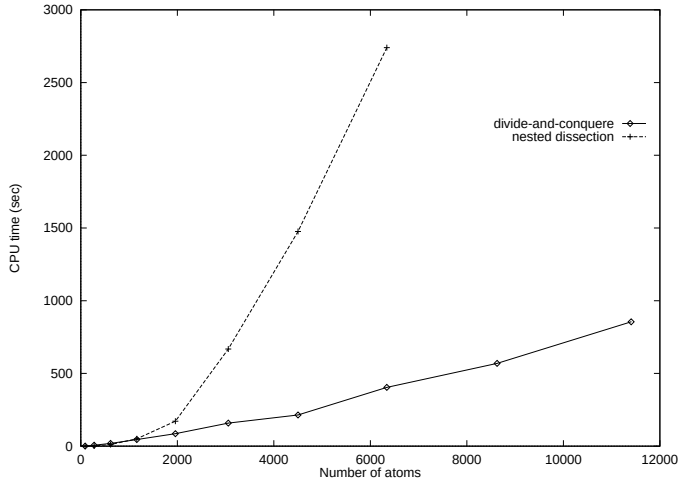

Figure 3: Total CPU times (sec) of the gradient transformation on diamond fragments.

4.5 Geometry optimizations

In order to be sure that the newly developed coordinate transformation algorithms work well in the context of geometry optimization, large diamond fragments have been optimized in the complete set of primitive internal coordinates, using the steepest descent method. Fig (4) shows how the maximum forces converge. The smooth convergence can be achieved only after projecting out redundancies from the internal coordinate displacements. Similar optimizations have been carried out on water clusters: forces and energies decrease smoothly.

4.6 Biomolecules

Table (1) contains gradient transformation data for biomolecules. Two methods are compared. The abbreviation ND stands for the method previously developed for quasi-1D systems, using the nested dissection ordering, while PD denotes the divide-and-conquer technique, based on prism distributions, described in the present paper. For the molecules which are not surrounded by water, the ND technique proved to be more efficient. In the PD

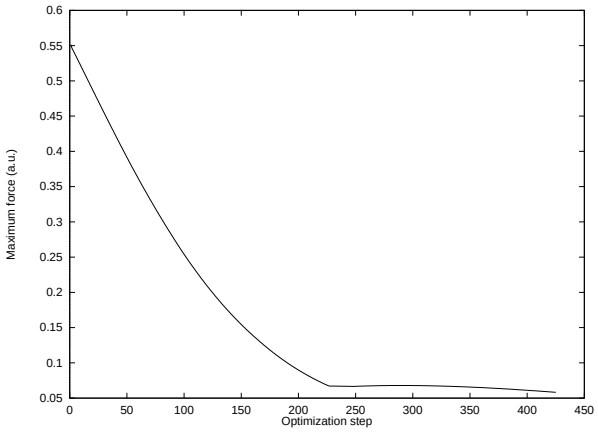

Figure 4: Maximum force on the internal coordinates during the geometry optimization of a 1162 atom diamond fragment.

method a considerable amount of skeletal coordinates should be introduced: approximately $3N$ for each of the six prism distribution. This means that the total size of the linear system of equations is supplemented approximately by $6 \times 3N$. Due to the quasi-1D nature of these biopolymers, the ND method gives already satisfactory results.

For the molecule 4BLM the ND technique needs quite a large number of iteration steps. After the introduction of additional internal coordinates the number of iteration steps may strongly decrease. For example, after the addition of the skeletal coordinates the number of iteration steps decreases to 2.

For the biomolecules in water environment skeletal coordinates have been used for both ND and PD methods. The introduction of skeletal coordinates was necessary even for the ND technique, in order to connect disjunct parts of the system (water molecules among each other and with respect to the protein). Here, the divide-and-conquer PD technique proved to be clearly superior, it requires much less memory and CPU time than the ND method.

5 Summary

In this article we describe a divide-and-conquer technique for carrying out the coordinate transformation problem of molecular geometry optimization. The memory requirements of the new technique scale linearly with system size, even for massively three-dimensional networks of atoms. The CPU time required for the coordinate transformations scales near linearly with the system size, when an appropriate choice of internal coordinates is available. To ensure this condition, in addition to the local primitive internal coordinates skeletal coordinates have been introduced. A new projection technique has been developed, to separate

Table 1: Data of the gradient trasformation of some biomolecules. Molecules are denoted by their PDB entry. The method "ND" denotes nested dissection ordering without the divide and conquer technique. "PD" denotes the inclusion of the divide and conquer technique into the coordinate transformation. The numbers after the specification of the biomolecule describe the radius (Å) of the spherical water cluster into which the biomolecule is placed. N: number of atoms; N_i: number of internal coordinates; N_{Ch}: number of non-zero matrix elements in the Cholesky factor; Δt_{Gr}: CPU time of one step (sec); N_{Gr}: number of iteration steps; Δt_T: total CPU time of the gradient transformation (sec).

Molecule	Method	N	N_i	N_{Ch}	Δt_{Gr}	N_{Gr}	Δt_T
1CBN	ND	642	3396	64098	0.7	5	3.7
1BPI	ND	898	4993	101939	1.2	5	5.8
1UOX	ND	4684	26030	595069	10.1	19	192.8
4BLM	ND	4023	22349	519570	9.9	55	547.0
1CBN	PD	642	15714	191103	6.5	11	72.1
1BPI	PD	898	22219	256881	9.4	12	113.0
1UOX	PD	4684	115472	1496835	45.7	23	1051.4
4BLM	PD	4023	98363	1588219	59.1	12	709.4
1CBN-18	ND	3087	70929	8661008	1188.6	2	2377.3
1BPI-18	ND	2191	51366	5622900	643.7	2	1287.5
1BPI-22	ND	4003	92247	13466976	2225.5	2	4451.1
1CBN-18	PD	3087	70929	1022019	38.9	10	389.5
1BPI-18	PD	2191	51366	720157	26.2	11	288.1
1BPI-22	PD	4003	92247	1352478	48.3	12	579.5

the redundant and non-redundant parts of the displacements of the internal coordinates. This technique proved essential in improving the convergence of the geometry optimization. Parallelization of the Cholesky factorization is feasible, when non-interacting prisms are considered. Also the projection of the redundancies can be parallelized, since the SVD problems can be solved separately and the rest consists only of matrix multiplications. The algorithms described in this paper are especially useful when the conformation of biomolecules in solvent is studied, or when the effects of dopings or lattice defects destroy the translational symmetry of three-dimensional networks of atoms.

6 Acknowledgement

K.N. is gratefully indebted to Prof. Jean-Louis Rivail for his hospitality at Université Henri Poincaré, Nancy I, and for his continuous support and interest in our work. The authors thank François Dehez, Christophe Chipot and Aylin Sungur for providing us with some of

the coordinates of the test molecules. K.N. gratefully acknowledges a postdoctoral fellowship to the Center Charles Hermite.

References

[1] S. Goedecker, Rev. Mod. Phys. **71**, 1085 (1999).

[2] G. Scuseria, J. Phys. Chem. **103**, 4782 (1999).

[3] Ö. Farkas and H. Schlegel, J. Chem. Phys. **109**, 7100 (1998).

[4] S. Billeter, A. Turner, and W. Thiel, Phys. Chem. Chem. Phys. **2**, 2177 (2000).

[5] B. Paizs, G. Fogarasi, and P. Pulay, J. Chem. Phys. **109**, 6571 (1998).

[6] B. Paizs, J. Baker, S. Suhai, and P. Pulay, J. Chem. Phys. **113**, 6566 (2000).

[7] K. Kudin, G. Scuseria, and B. Schlegel, J. Chem. Phys. , in press.

[8] K. Németh, O. Coulaud, G. Monard, and J. G. Ángyán, J. Chem. Phys. **113**, 5598 (2000).

[9] J. Millam and G. Scuseria, J. Chem. Phys. **106**, 5569 (1997).

[10] M. Challacombe, J. Chem. Phys. **110**, 2332 (1999).

[11] M. Benzi, C. Meyer, and M. Tuma, SIAM J. Sci. Comput. **17**, 1135 (1996).

[12] M. Eliashevich, Compt. Rend. Acad. Sci. U.S.S.R. **28**, 605 (1940).

[13] J. E. Wilson, J. Chem. Phys. **9**, 76 (1941).

[14] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems* (Prentice Hall, Englewood Cliffs, NJ, 1981).

[15] N. J. Higham, in *Reliable numerical computation*, edited by M. Cox and S. Hammarling (Clarendon Press, Oxford, 1990).

[16] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne, Nucleic Acids Research **28**, 235 (2000), http://www.rcsb.org/pdb.

[17] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria and M. A. Robb *et al.*, GAUSSIAN 98, revision A.7, 1998.

[18] S. Mayo, B. Olafson, and W. Goddard, J. Phys. Chem. **94**, 8897 (1990).

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399